 Print

**View Reviews**

**Paper ID**

10

**Paper Title**

Resource Over-commitment for HPC Cloud with Quality-of-Service Guarantees

**Reviewer #1**

**Questions**

* **1. Please rate the paper**
  + Accept
* **2. Reviewer's Familiarity with the Subject Area [Private Field Visible to the track chairs]**
  + Medium
* **3. Technical Contribution**
  + Medium
* **4. Originality**
  + Medium
* **5. Presentation**
  + Excellent
* **6. Relevance to the Virtualization (VIRT) track**
  + High
* **7. Please justify your rating and provide detailed feedback to the authors.**
  + The paper describes optimizing resource allocation and management with VMware ESXi hypervisor, and shows experiments for CPU and memory over-commitment. Topic is not new, but the paper has a practical view with an expectation of the generalization in the approach going beyond the testbed experiments. The include resources are limited that storage and network workloads are also important in the resource management. The paper indicated them as the future work. The results show what the paper claims.

**Reviewer #2**

**Questions**

* **1. Please rate the paper**
  + Weak Accept
* **2. Reviewer's Familiarity with the Subject Area [Private Field Visible to the track chairs]**
  + Medium
* **3. Technical Contribution**
  + Low
* **4. Originality**
  + Medium
* **5. Presentation**
  + Good
* **6. Relevance to the Virtualization (VIRT) track**
  + High
* **7. Please justify your rating and provide detailed feedback to the authors.**
  + This paper shows the viability overcommitting CPU and memory when of using clusters of VMs to run HPC jobs. The main contribution of this paper is the usage of dynamic VM migration whenever the aggregate memory requirements for the VMs running on a node exceeds the memory capacity of that node.  
      
    The authors present experimental results that show the viability of the proposed approach. The authors do not specify whether they use single-process or multi-process jobs in their experiments, but from the text (and the results) it seems that only single-process jobs are scheduled. This would make the results less valuable, since most HPC workloads are multi-process/multi-node and the inter-process communication (e.g., MPI or SHMEM) tends to be in the critical path.  
      
    In my opinion, the main weakness of this paper is that it ignores the provisioning of network resources. Are NICs in the nodes shared by all VMs? (Given the hardware configuration, this seems to be the case). What is the impact of sharing the NIC across VMs? For instance, how is the latency/throughput of collective operations affected? The lack of this analysis raises some doubts about the claim that clusters of VMs with over-commitment of hardware resources is a practical architecture to run HPC workloads.

**Reviewer #3**

**Questions**

* **1. Please rate the paper**
  + Weak Accept
* **2. Reviewer's Familiarity with the Subject Area [Private Field Visible to the track chairs]**
  + High
* **3. Technical Contribution**
  + Medium
* **4. Originality**
  + Low
* **5. Presentation**
  + Fair
* **6. Relevance to the Virtualization (VIRT) track**
  + High
* **7. Please justify your rating and provide detailed feedback to the authors.**
  + Summary:  
    The authors demonstrate the potential benefits of using resource over-provisioning and live migration in a throughput-oriented HPC setting. They first examine CPU over-commitment using virtual clusters, then both CPU and memory, and find that in both cases the use of virtualization is justified.  
      
    General comments:  
    Overall I thought this was a good paper, though it is not too surprising that throughput-oriented workloads match well with virtualized environments. I think the paper will generate interesting discussions in the conference. However, the elision of key details on some experiments leave the reader with insufficient information to interpret some results properly.  
      
    Details:  
    \* Figure 2 seems to indicate that certain VMs have affinity to certain physical nodes (i.e. the login VM should be placed on the login physical node). Physical differences between compute nodes and I/O nodes would also suggest that this is what you’d want to do. However, I don’t see any mention of how this plays into the live migrations that DRS can perform. Is it only compute node <-> compute node? Or is it agnostic?  
    \* It would be nice to have a better idea of what workload characteristics are contributing to CPU utilization. e.g. how much IO is bioperf doing (if any)?  
    \* Would like to see figure 7 broken down into how much of the time is spent in live migration overheads.   
    \* What kind of live migration mechanisms is DRS using? Pre-copy? post-copy?  
    \* You don’t say how many logical threads you have per core on either cluster. This makes it difficult to interpret the results in Figure 4 and the claims attached to it.   
    \* It is unclear why CPU utilization is a good measure of fairness (fig 5) when in the previous paragraphs you indicate that you’re measuring the “CPU utilization of all VMs.” It’d be better to have a more rigorous metric for fairness. E.g. CJET without relative to CJET with competing VCs.  
    \* A big potential issue here is workload interference. You might be missing pathological cases by looking at your performance numbers only in the aggregate. The scheduler should avoid, e.g. co-locating two memory-intensive VCs.  
    \* Information leakage is also a concern here. I doubt the NNSA labs would want to see their simulation codes running co-located with other users 🙂. Though I do believe you scope to private cloud.  
    \* I’m not sure it’s sound to extrapolate tail latency when you’re not incorporating queueing delays (unless you are including that in the individual job execution time numbers in fig 9).  
      
    Nitpicks:  
    \* The paper could use a grammar/spelling check pass  
    \* define VC acronym before use

**Reviewer #4**

**Questions**

* **1. Please rate the paper**
  + Reject
* **2. Reviewer's Familiarity with the Subject Area [Private Field Visible to the track chairs]**
  + High
* **3. Technical Contribution**
  + Low
* **4. Originality**
  + Low
* **5. Presentation**
  + Good
* **6. Relevance to the Virtualization (VIRT) track**
  + Medium
* **7. Please justify your rating and provide detailed feedback to the authors.**
  + The papers focus on the resource allocation problem for HPC cloud with multi-tenancy. The authors initially stated that they designed CPU and memory over-commitment methodology to improve resource utilization and system throughput, although I am afraid that they did not really achieve their objective regarding QoS. Rather than effectively designing a mechanism, they evaluated scenarios and drew conclusions on the benefits of overcommitting both CPU and memory. However, there were cases in which execution failed due to the lack of enough resources (memory), and no solution was proposed. Yet, dynamic VM migration was held but no careful analysis was presented.   
      
      
    You say that:” ...Studies have demonstrated that the performance gap between virtual and bare-metal for HPC throughput workloads has closed, with just 1 or 2 percentage difference”  
    --> This is questionable. You should provide some information on the workload main characteristics and VM type under consideration.   
    --> how is the shares-based mechanism in the scheduler of VMware ESXi?  
      
    --> There are various works available in the literature that tackles vertical memory elasticity. The authors should analyse the proposed mechanisms and or either apply them or discuss the advantages or disadvantages.   
      
    --> it seems that the efficiency of the proposed approach relies on the assumption that the physical cluster’s memory capacity has enough memory for the tenant’s requirements.   
      
    --> the details of the approach are not really presented in the paper.  
      
    Regarding the results shown in Figure 3  
    --> when 1 VC is used, the execution concurrency is limited by torque?   
    --> why is there no difference between 2VCs and 4VCs?  
    --> 1, 2 and 4 VCs are established in 20 nodes?   
    --> Given a set of jobs/tenants, can we derive how many VC clusters should be deployed?   
      
    Regarding the results shown in Figures 3 and 4, is the number of jobs the same for 1, 2 and 4 VCs?  
      
    “the theoretical maximum utilization across the 16 20-core nodes can be calculated as 16 \* 20 \* 100% = 32,000%. Apparently, Figure 4 reveals that HyperThreading and Turbo are able to improve CPU utilization beyond above maximum value.”  
    --> you should also show what happens if these parameters are not set  
      
    --> When you have 1VC, CPUs are not shared, are they? But, on average, how much CPU % do the jobs need?   
      
    You also say:   
    “Specifically, each tenant can negotiate an appropriate share of the physical cluster before VC is allocated.”  
    --> but how is this done in your proposed framework/system? Does each tenant provide the share amount? You also say that the ESXi scheduler allocates CPU based on the ratio of shares among all the running VMs. So, are these shares stipulated at run time by the ESXi scheduler?  
      
    In scenario 3 (memory demanding), two simultaneous streams of HPCC jobs did not finished due to the lack of enough memory  
    --> so, how are you guaranteeing QoS?  
      
    --> In the last paragraph of the Experiments Section the authors mention about migration due to intensive memory stress. However, it is not clear what is the mechanism/condition that triggers the migration events, which kind of migration is used, and if the migration events held were really necessary.

**Reviewer #5**

**Questions**

* **1. Please rate the paper**
  + Weak Accept
* **2. Reviewer's Familiarity with the Subject Area [Private Field Visible to the track chairs]**
  + High
* **3. Technical Contribution**
  + High
* **4. Originality**
  + Medium
* **5. Presentation**
  + Good
* **6. Relevance to the Virtualization (VIRT) track**
  + High
* **7. Please justify your rating and provide detailed feedback to the authors.**
  + The goal of this work is to optimize resource allocation for HPC cloud which is currently static in nature. CPU and memory over-commitment schemes are proposed to improve resource utilization and system throughput. Authors achieve this goal by offering each tenant a virtual cluster of VMs that mimic the underlying physical cluster. And then simultaneously run multiple virtual clusters on a physical cluster and delegate the resource management task to the hypervisor. This work focuses on hypervisor based full virtualization (VMware ESXi) and the hypervisor is responsible for providing SLA guarantees to each tenant and assigning idle resources from one tenant to another.  
      
    Each tenant gets one virtual cluster to run their jobs/tasks. In the virtual cluster the tenant has one VM per node. And CPU over-commitment is achieved by assigning vCPUs equal to the number of physical cores in the node. Thus, each tenant VM has the illusion that it has all the resources of the node. When multiple tenant’s VMs run on the same physical node, QoS guarantees and changing VM vCPU count are done by a share-based CPU scheduler in the hypervisor. Dynamic VM migration is used to relax the memory pressure created by memory over-commitment on a particular node as the VM is migrated to another node in the cluster to spread the memory load.   
      
    Strengths -  
      
    1. CPU over-commitment scheme applied by authors is able to improve performance, CPU utilization of the cluster, and have fairness among virtual clusters.  
      
    2. Authors perform a feasibility test of their idea of both CPU and memory over-commitment.  
      
    3. The work shows that without memory reclamation and no VM migration, jobs fail when memory is over-committed. But when both techniques are used it improves the performance of the system despite VM migrations, achieves 2x memory over-commitment ratios, and p99 latency guarantees.  
      
    Comments/Weakness –  
      
    1. Authors perform experiments upto 4x over-commitment of CPU. It would have been nice to see the limit of CPU over-commitment in a node or at what over-commitment ratio does performance starts to degrade below acceptable levels (like 16x, 32x, etc.).  
      
    2. The paper claims to outline best practices to be followed for resource over-commitment but no guidelines have been laid out about deciding the over-commitment ratio or how to choose/decide the optimal number of virtual clusters to run per node.  
      
    3. Authors compare their CPU over-commitment scheme to bare-metal but do not compare their system to bare-metal nodes when both CPU and memory are over-committed. They do state that they were not able to do this due to testbed limitations.

**Reviewer #6**

**Questions**

* **1. Please rate the paper**
  + Reject
* **2. Reviewer's Familiarity with the Subject Area [Private Field Visible to the track chairs]**
  + High
* **3. Technical Contribution**
  + Low
* **4. Originality**
  + Low
* **5. Presentation**
  + Good
* **6. Relevance to the Virtualization (VIRT) track**
  + High
* **7. Please justify your rating and provide detailed feedback to the authors.**
  + This article studies the use of virtual clusters (clusters of VMs) for running HPC workloads while over-committing resources. The idea is to schedule VMs of multiple tenants on the same cores to maximize utilization even when some tenants are idle. Over-committing memory is also studied. In that case, the authors rely on vSphere's scheduler to migrate VMs depending on their current memory usage to try to prevent swapping. The hope is that at any given point the total sum of memory used by all VMs can fit within the total memory of the cluster. There is however no guarantee which can lead to unexpected crashes.  
      
    While it was interesting to see techniques such as VM migration successfully applied to memory intensive computations, I was a bit disappointed by the content of the paper compared to what is promised in the title and abstract:  
      
    At first glance the paper seems to address generic HPC workloads, and nothing suggests otherwise in the title or the abstract. However, its results are only applicable to a small subset of HPC workloads that is more often referred to as throughput computing than as HPC: large batches of independent and single threaded tasks. The first mention that the paper targets throughput computing comes in the middle of the evaluation section. Studying tightly coupled multithreaded and parallel (MPI) applications which make the bulk of what is most commonly referred to as HPC workloads is left for future work.  
      
    The authors claim in the abstract that they achieve performance improvements over bare metal clusters in certain scenarios. However the paper later explains that this is due to using both hyperthreads for scheduling the VMs vCPUs while not using only one hyperthread for scheduling the bare metal tasks. Running bare metal tasks on both hyperthreads cancels this advantage.  
      
    The title of the paper suggests QoS guarantees. However, in practice, the authors measure the range of execution times of a specific set of benchmarks running on overcommitted resources. This allows them to provide guarantees for this specific scenario but not to provide more general insights into the slowdowns that would be incurred by other applications. I was also surprised that the paper never clearly discusses problems such as noisy neighbors and techniques to mitigate these issues. I would also have liked to see a few words about the security implications of sharing cores or hyperthreads such as side-channels attacks.

**Reviewer #7**

**Questions**

* **1. Please rate the paper**
  + Weak Accept
* **2. Reviewer's Familiarity with the Subject Area [Private Field Visible to the track chairs]**
  + Medium
* **3. Technical Contribution**
  + Medium
* **4. Originality**
  + Low
* **5. Presentation**
  + Excellent
* **6. Relevance to the Virtualization (VIRT) track**
  + High
* **7. Please justify your rating and provide detailed feedback to the authors.**
  + The paper is very well written, the experiments are carefully executed and, as far as I can see, technically sound. I think it would make a decent contribution to the conference.  
      
    My main concern is that very little is unexpected in the results. This is especially true for the CPU-overcommitment experiments. In these experiments the worlkload is a set of CPU-intensive threads, and all available CPU logical cores are available for all VMs. In this setting, the hypervisor is essentially in the same scenario as a kernel scheduling a set of threads, which is hardly a new problem. Unsurprisingly, what we observe is that proportional-share schedulers work essentially the same whether they are scheduling standard procesess or vCPUs. This is essentially what one gets from Sections 4.2-3.  
      
    The experiments on memory overcommitment are more interesting because of the possibility of using VM migration as a load-balancing mechanism. While this has been studied extensively, it has always been a problem to implement migration for standard processes in commodity OSes. The experiments described in this part of the paper involve a large set of resources and realistic workloads, and make for an interesting read.